(84) **États désignés** *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé : Procédé pour réaliser une migration de connexions dans une architecture multi-ordinateurs (cluster), depuis un premier nœud, appelé nœud primaire ou opérationnel, comprenant un premier ordinateur dudit cluster sur lequel une application logicielle initiale est exécutée, vers au moins un nœud secondaire comprenant un autre ordinateur dudit cluster. Ce procédé met en œuvre une adresse réseau virtuelle qui est portée par le premier ordinateur et qui est transférée sur l'autre ordinateur, cette adresse réseau virtuelle étant prévue comme lien de dialogue entre le cluster et des ordinateurs clients connectés au cluster et concernés par l'application logicielle. Les connexions concernées peuvent par exemple être associées à une application logicielle destinée à être répliquée sur un autre ordinateur de façon à permettre un basculement de service de l'application initiale vers sa réplique.

«Procédé de migration de connexions dans une architecture multi-ordinateurs, procédé pour réaliser une continuité de fonctionnement mettant en œuvre ce procédé de migration, et système multi-ordinateurs ainsi équipé»

La présente invention concerne un procédé pour réaliser la migration de connexions dans une architecture multi-ordinateurs. Elle vise également un procédé pour réaliser une continuité de fonctionnement d'une application logicielle dans une architecture multi-ordinateurs (cluster), mettant en œuvre ce procédé de migration, ainsi qu'un système multi-ordinateurs implémentant ce procédé de continuité de fonctionnement.

Le domaine de l'invention est celui des clusters d'ordinateurs formés de plusieurs ordinateurs collaborant entre eux. Ces clusters sont par exemple utilisés pour exécuter des applications logicielles. Ainsi, à un instant donné, une application est exécutée sur l'un des ordinateurs du cluster, appelé nœud primaire ou opérationnel (OP), tandis que les autres ordinateurs du cluster sont appelés nœuds secondaires ou « stand-by » (SB), dans un contexte d'architecture redondante.

Or, l'exploitation de tels clusters montre que se posent des problèmes de fiabilité qui peuvent être dus à des défaillances du matériel ou du système d'exploitation, à des erreurs humaines, ou à la défaillance des applications elles-mêmes.

Pour résoudre ces problèmes de fiabilité, il existe actuellement des mécanismes, dits de haute disponibilité, qui sont mis en œuvre sur la plupart des clusters actuels et qui sont basés sur un redémarrage automatique à froid de

l'application sur un nœud de secours parmi l'un des nœuds secondaires du cluster.

Or ces mécanismes basés sur un redémarrage automatique ne permettent pas d'assurer une continuité totale du service fourni par l'application en cours d'exécution au moment de la défaillance.

Se pose en particulier le problème complexe de la migration des connexions de réseau qui doit être résolu au moment du basculement de service entre deux ordinateurs d'un cluster.

Un premier objectif de la présente invention est d'apporter une solution à ce problème en proposant un procédé pour réaliser une migration de connexions dans une architecture multi-ordinateurs (cluster), depuis un premier nœud, appelé nœud primaire, comprenant un premier ordinateur dudit cluster sur lequel une application logicielle initiale est exécutée, vers au moins un nœud secondaire comprenant un autre ordinateur dudit cluster.

Ce premier objectif est atteint avec un tel procédé de migration mettant en œuvre une adresse réseau virtuelle qui est portée par le premier ordinateur et qui est transférée sur l'autre ordinateur, ladite adresse réseau virtuelle étant prévue comme lien de dialogue entre le cluster et des ordinateurs clients connectés audit cluster et concernés par l'application logicielle.

Dans un mode de réalisation avantageux, Les messages en provenance d'un client sont capturés avant la prise en compte par la couche réseau du cluster. En particulier, lorsque ce procédé de migration est mis en œuvre dans le contexte d'un protocole TCP/IP, la capture des messages est effectuée au niveau des tables « IP ».

La migration de connexions peut être aussi appliquée, au delà de la tolérance aux pannes, à la mobilité des réseaux:

re-connexion d'un ordinateur portable ou autre appareil communiquant mobile d'un réseau physique à un autre, sans perte des connexions et contextes applicatifs, alors que les solutions existantes mettent en oeuvre un serveur intermédiaire non nécessaire avec le procédé de migrations selon l'invention.

Par rapport aux travaux de recherche antérieurs sur la migration des connexions, le procédé décrit selon l'invention présente les avantages suivants :

- il ne nécessite pas de modification du protocole de transport TCP, malgré les limitations que présente celui-ci,
- par voie de conséquence, les machines distantes ne sont impactées en aucune manière lors de la mise en oeuvre du procédé,
- l'implémentation de la migration de connexions ne nécessite pas de modification du système d'exploitation, mais simplement le chargement d'un module noyau dynamique indépendant.

Les protocoles de transport basés sur la couche « socket IP-UDP » sont aussi automatiquement pris en compte, quelles que soient leurs caractéristiques (exemple: protocoles de streaming audio ou video).

Il est important de noter que le procédé de migration de connexions selon l'invention peut prendre en compte une pluralité de nœuds secondaires ou « stand-by », procurant ainsi un effet multi-échelle (« scalabilité »).

Le procédé de migration selon l'invention peut être avantageusement mais non limitativement mis en œuvre pour la migration de connexions qui sont associés à une application logicielle destinée à être répliquée sur un autre ordinateur

de façon à permettre un basculement de service de l'application initiale vers sa réplique.

Le procédé de migration de connexions selon l'invention peut aussi être mis en œuvre de façon complètement autonome, indépendamment de situations de basculement d'une machine à une autre ou de réplication d'applications logicielles.

On peut ainsi prévoir d'appliquer le procédé de migration selon l'invention pour une optimisation automatique de ressources informatiques par partage de charge par répartition dynamique de processus. Ce procédé de migration peut aussi être appliqué pour une maintenance non interruptive par relocation à la demande de processus au travers d'un réseau de ressources informatiques, ou pour une préservation de contexte applicatif dans des applications mobiles.

Un autre but de la présente invention est de proposer un procédé pour réaliser une continuité de fonctionnement d'une application logicielle dans une architecture multi-ordinateurs (cluster), cette application étant exécutée à un instant donné sur l'un des ordinateurs du cluster, appelé nœud principal, les autres ordinateurs dudit cluster étant appelés nœuds secondaires, ce procédé mettant en œuvre le procédé de migration de connexions selon l'invention.

Cet autre objectif est atteint avec un procédé comprenant les étapes suivantes :

- maintien au fil de l'eau d'au moins un clone de l'application sur au moins un des nœuds secondaires,
- en cas de détection d'une défaillance ou d'un événement affectant ledit nœud principal, basculement de service vers l'un au moins desdits clones, et
- migration de connexions réseau.

Ainsi, avec le procédé de migration selon l'invention, il est désormais possible, grâce à la migration des

connexions réseau, de rendre les basculements de service vers des clones, transparents pour le monde extérieur communiquant avec l'application.

Par ailleurs, le mécanisme de migration de connexions mis en œuvre dans le procédé de migration selon l'invention n'implique pas une modification du code source de l'application et est donc non intrusif dans l'application, à la différence des procédés de migration de l'art antérieur.

Les clones mis en œuvre dans le procédé de continuité de fonctionnement selon l'invention sont dits « chauds », c'est-à-dire qu'ils sont la réplique exacte de l'application et de tout son contexte opératoire. Ils sont mis à jour régulièrement (périodiquement ou sur évènements caractéristiques). Ces clones contiennent toutes les ressources et informations requises par l'application pour fournir son service.

Le procédé de continuité de fonctionnement selon l'invention permet en outre de superviser l'état de toutes les ressources nécessaires au bon fonctionnement de l'application. Quand la dégradation rédhibitoire de l'une d'entre elles est détectée, le procédé de continuité de fonctionnement selon l'invention prévoit une élection d'un clone comme nouveau primaire et lui ordonne de prendre la main.

Cette élection est appelée basculement et est transparente pour le reste du monde qui communique avec l'application : bien que le nœud primaire soit mis hors service, le service fourni par l'application n'est pas interrompu car il est repris avec tout son contexte par le clone élu.

On peut ainsi garantir que tout message transmis par le reste du monde à l'application sera traité, soit par le primaire (pré-basculement), soit par le clone (post-

basculement). Pour ce faire, le procédé de continuité de fonctionnement selon l'invention peut en outre comporter un enregistrement sur chaque clone (en plus du mécanisme de clonage périodique), de tous les messages reçus par le primaire depuis la dernière mis à jour des clones. Ces messages seront réinjectés dans le clone élu nouveau primaire en cas de basculement.

La réplication holistique, pour être complète, inclut la réplication de ressources « noyau », comme par exemple l'état des piles protocolaires mis en œuvre pour gérer la connectivité de l'application à protéger avec le monde extérieur (ses « clients »).

Un avantage important procuré par le procédé de migration selon l'invention est de rendre transparent pour les clients de l'application le basculement du service applicatif du primaire vers un secondaire. Techniquement, cela veut dire que les connexions établies par les clients avec l'application lors de son fonctionnement sur le primaire, doivent être transmises (migrées) vers les clones et ne doivent pas être rompues lors d'un basculement. Cette exigence est non triviale car pour les applications concernées par le procédé de continuité de fonctionnement selon l'invention, le monde extérieur (les clients) communique avec l'application essentiellement via des connexions TCP/IP, un protocole point à point « attaché » aux machines physiques sur lesquels résident applications et clients.

Le procédé de continuité de fonctionnement selon l'invention résout ce problème en implémentant un mécanisme de réplication de l'état de la pile protocolaire ainsi que d'un système d'enregistrement / re-jeu qui permet, suite à un basculement, de réinjecter les messages reçus par le nœud

primaire avant le basculement mais pas encore pris en compte par le clone.

L'état de la pile, dans le noyau du système d'exploitation, est périodiquement introspecté (analysé et capturé) sur la machine primaire, cet état étant transféré avec le point de reprise (checkpoint) holistique et restauré sur les nœuds secondaires.

En parallèle, tous les messages reçus par le nœud maître sont interceptés au plus bas niveau (avant d'être livrés à l'application sur le nœud primaire) et transférés pour être enregistrés sur les nœuds secondaires. Sur les nœuds secondaires, ces messages sont sauvegardés depuis le dernier point de reprise reçu.

En cas de basculement, le nœud secondaire élu prend la main en faisant tourner l'application à partir de son dernier point de reprise (ce point de reprise est légèrement dans le passé par rapport à l'application lors du basculement, puisque reçu périodiquement par les nœuds secondaires).

Pour ramener ce clone dans l'état présent, c'est à dire dans l'état de l'application au moment du basculement, les messages enregistrés sont réinjectés. En les rejouant, le clone nouveau primaire atteint alors l'état de l'application au moment du basculement. Ce re-jeu, dans certains cas, peut se faire de façon accélérée (compression du temps, élimination des « blancs »). Pendant ce re-jeu, les communications avec le monde extérieur sont fermées. Si de nouveaux messages sont reçus des clients pendant le re-jeu, ils sont refusés, mais sans déconnexion. Ce refus sera géré par le protocole (contrôle de flux) et sera vu par les clients comme un ralentissement du réseau ou du service.

Il faut noter que le re-jeu nécessite l'adjonction spécifique d'un canal d'injection de messages dans la file de réception du driver de l'interface réseau indépendamment du

niveau physique, le système d'émission de trames ne permettant pas le rebouclage (entrées-sorties half duplex) sur les interfaces physiques.

A l'issue du re-jeu, le clone est dans l'état exact de l'application avant le basculement et reprend la main en ré-ouvrant les communications avec le monde extérieur.

Il est à noter que dans certaines configurations et selon le but recherché, il est possible de mettre en œuvre la politique de basculement à chaud sur un clone, sans mettre en œuvre le re-jeu. Les impacts d'un tel scénario de reprise sont :

- la rupture des connexions en cours, donc, pour les clients connectés sur des sessions actives, niveau de protection inférieur à celui proposé par le re-jeu,

- la reprise à chaud (contexte applicatif préservé) immédiate (pas de temps de re-jeu pendant lequel les nouveaux messages sont temporisés), donc, ré-établissement plus rapide du service nominal avec acceptation plus rapide de nouveaux clients.

L'implémentation de l'une ou l'autre des ces politiques de reprise est un paramètre adaptable.

Il est à noter que pour la mise en œuvre du procédé de migration selon l'invention, on peut avantageusement utiliser des techniques d'ingénierie logicielle non intrusive dynamique, qui ont fait l'objet d'une demande de brevet publiée le 2 août 2002 sous le numéro FR2820221. Ces techniques d'ingénierie logicielle permettent de manipuler des applications dans leur représentation binaire (exécutable), de façon à rendre le procédé de réalisation de continuité de fonctionnement selon l'invention transparent pour l'application et donc générique.

Suivant un autre aspect de l'invention, il est proposé un système multi-ordinateurs prévu pour exécuter sur au moins

desdits ordinateurs au moins une application logicielle, implémentant le procédé pour réaliser une continuité de fonctionnement selon l'invention.

D'autres avantages et caractéristiques de l'invention apparaîtront à l'examen de la description détaillée d'un mode de mise en œuvre nullement limitatif, et des dessins annexés sur lesquels :

- la figure 1 illustre schématiquement un exemple de mise en œuvre du procédé de migration selon l'invention au sein d'un procédé de continuité de fonctionnement;
- la figure 2 illustre schématiquement un mécanisme de point de reprise (checkpointing) mis en œuvre dans un procédé de continuité de fonctionnement selon l'invention ; et
- la figure 3 illustre schématiquement des fonctions de supervision et de surveillance assurées sur les nœuds d'un système multi-ordinateurs (cluster) selon l'invention.

On va maintenant décrire, en référence aux figures précitées le fonctionnement du mécanisme de migration des connexions réseau mis en œuvre dans le procédé de migration selon l'invention.

La migration des connexions s'appuie sur l'utilisation d'une adresse réseau virtuelle dite adresse virtuelle du cluster. Cette adresse est portée par la machine qui détient l'application opérationnelle. Elle est transférée vers une machine SB lors du basculement (switch). Les clients doivent s'adresser à l'application clusterisée en dialoguant sur cette adresse virtuelle.

La capture des messages se fait avant la prise en compte par la couche réseau. Sur TCP/IP, cette capture est faite au niveau des "IP Tables" se qui assure la portabilité.

Les messages reçus sur l'adresse IP virtuelle assignée au cluster sont émis vers le ou les machines SB sur un canal de type multicast (diffusion simultanée en direction de plusieurs destinataires) fiable qui s'assure de la transmission des paquets. Lorsque le message est reçu sur toutes les machines SB, celui ci est transmis à la couche réseau du nœud opérationnel OP. Sinon le message est supprimé (la couche transport du distant assurant la retransmission).

Ce mécanisme permet de garantir qu'un message peut être rejoué sur une machine SB lorsqu'il est pris en compte sur le nœud opérationnel OP. Le filtrage "IP Tables" permet de ne s'intéresser qu'au message concernant l'adresse virtuelle du cluster.

Lors d'une copie (dump), une marque de copie (dump) est émise vers les machines SB afin de dater la copie dans le journal (log). Ainsi on sait à partir de quel paquet il faut commencer le re-jeu lors d'un basculement (switch).

Le module "IP tables" est un module noyau indépendant de la couche TCP/IP qui est chargé dynamiquement. Aucune modification de la pile TCP/IP n'est requise ni sur le cluster ni sur les machines distantes.

Les appels systèmes génériques *getsockopt* et *setsockopt* sont étendus, via un pilote (driver), pour capturer/modifier les paramètres *socket* suivants:

- ports local & distant
- numéro de référence local & distant
- numéro du prochain paquet à émettre & attendu
- horloge (« timer ») d'émission & de réception
- taille de fenêtre (« window size »)

etc.

La sauvegarde de l'état de la « socket » prend également en compte la liste des paquets en attente d'émission (send

queue), ceux qui sont en transit (émis mais non acquittés), et les paquets reçus mais pas encore lus par l'application (file de réception : « receive queue »).

La sauvegarde est faite dans le contexte du processus au moment du dump, après l'émission de la marque de log. Ce mécanisme assure que tous les paquets seront rejoués. Si un paquet est reçu entre l'émission de la marque de log et la capture de l'état de la « socket », celui-ci est automatiquement ignoré par la couche transport lors du basculement (switch). Ce traitement est un fondement de la couche transport.

L'extension des appels système *getsockopt* et *setsockopt* est faite par un module noyau chargé dynamiquement qui ne requiert aucune modification du code source du noyau.

Le procédé de migration de connexions selon l'invention peut être directement intégré au sein d'un procédé de continuité de fonctionnement pour un cluster de machines, en référence à la figure 1. Un tel système met en œuvre une supervision et une détection de défaillance, un mécanisme de migration de process incluant un mécanisme de point de reprise (checkpoint), un mécanisme de migration de connexions selon l'invention, et un gestionnaire de ressources système. Ce procédé de continuité de fonctionnement inclut aussi une fonction d'introspection de ressources, un mécanisme de réplication du système de fichiers, un mécanisme d'édition et de mise à jour d'un journal des événements, et un mécanisme de re-jeu.

On va maintenant décrire le mécanisme de commutation mis en œuvre dans une migration des connexions réseau au sein du procédé de migration selon l'invention.

Lorsqu'une machine SB devient un nœud opérationnel OP, l'adresse IP virtuelle est créé sur le nouveau nœud opérationnel OP mais les règles de filtrage interdisent toute

arrivée de message de l'extérieur pendant la phase de jeu. Une fois le jeu terminé, les règles de filtrage sont modifiées afin de dispatcher les messages sur les machines SB.

Si des messages sont émis par le site distant, ils seront détruits à l'arrivée dans la machine nouvellement OP, et seront retransmis par la couche distante sur « timeout » (hors temps). Ce mécanisme est un des fondements de la couche transport.

La recréation des « sockets » (couche de communication) se fait en 2 phases:

- avant re-jeu vers un « loopback » (dispositif de stockage virtuel),
- après re-jeu en les reconnectant vers l'extérieur.

Avant le re-jeu, les « sockets » sont connectés à un « loopback » qui permet la réinjection des paquets enregistrés depuis la dernière restauration.

Les paramètres de la « socket » sauvegardés lors de la précédente copie « dump » sont remis en place via l'appel étendu *setsockopt*.

Les paquets enregistrés depuis la dernière restauration sont émis vers la couche transport comme s'il avait été reçus directement d'un équipement distant (clients). Les messages transmis par la couche transport sont automatiquement détruits afin de ne pas perturber le distant.

A la fin du re-jeu, l'état de la « socket » correspond à celui sur le nœud opérationnel OP avant le basculement (switch). Il est alors possible de reprendre le dialogue avec le distant. Cette reprise se fait en modifiant les paramètres « sockets » concernant l'adresse du distant, et en modifiant les règles de filtrage pour autoriser l'entrée et la sortie

de message sur l'adresse virtuelle du cluster. Le dialogue reprend alors normalement.

Le procédé de migration de connexions selon l'invention peut être mis en œuvre dans le cadre d'une interaction entre un serveur opérationnel et un serveur miroir tel qu'illustré par la figure 2, dans lequel un mécanisme de point de reprise (checkpoint) est activé, avec génération de copies périodiques incrémentales.

Pour la mise en œuvre d'un procédé de continuité de fonctionnement selon l'invention, entre un nœud opérationnel d'un cluster et un ou plusieurs nœuds secondaires de ce cluster, une base MIB (Management Information Base) est accédée par des pilotes d'introspection et de surveillance et par des commandes du cluster, en référence à la figure 3. Cette base MIB intervient dans la gestion du cluster sur le nœud opérationnel et sur des nœuds de « back-up » pour les fonctions de restauration et de point de reprise, de décision de basculement et d'organisation de ce basculement. Un gestionnaire de supervision alimenté par la base MIN assure des fonctions de contrôle et de MIB synthétique, en relation avec une interface utilisateur graphique GUI.

Bien sûr, l'invention n'est pas limitée aux exemples qui viennent d'être décrits et de nombreux aménagements peuvent être apportés à ces exemples sans sortir du cadre de l'invention.

REVENDICATIONS

1. Procédé pour réaliser une migration de connexions dans une architecture multi-ordinateurs (cluster), depuis un premier nœud, appelé nœud primaire, comprenant un premier ordinateur dudit cluster sur lequel une application logicielle initiale est exécutée, vers au moins un nœud secondaire comprenant un autre ordinateur dudit cluster, caractérisé en ce qu'il met en œuvre une adresse réseau virtuelle qui est portée par ledit premier ordinateur et qui est transférée sur ledit autre ordinateur, ladite adresse réseau virtuelle étant prévue comme lien de dialogue entre le cluster et des ordinateurs clients connectés audit cluster et concernés par l'application logicielle.

2. Procédé de migration selon la revendication 1, dans lequel les connexions sont associés à une application logicielle destinée à être répliquée sur au moins un autre ordinateur de façon à permettre un basculement de service de l'application initiale vers sa réplique.

3. Procédé de migration selon l'une des revendications 1 ou 2, caractérisé en ce que les messages en provenance d'un client sont capturés avant la prise en compte par la couche réseau du cluster.

4. Procédé de migration selon la revendication 3, mis en œuvre dans le contexte d'un protocole TCP/IP, caractérisé en ce que la capture des messages est effectuée au niveau des tables « IP ».

5. Procédé de migration selon l'une des revendications précédentes, caractérisé en ce que les messages reçues sur l'adresse réseau virtuelle sont émis vers le ou les ordinateurs secondaires sur un canal de type multicast.

6. Procédé de migration selon les revendications 4 et 5, caractérisé en ce qu'il comprend une capture et une modification des paramètres « socket », via des appels systèmes génériques étendus.

7. Procédé de migration selon la revendication 6, caractérisé en ce que les paramètres « socket » capturés et modifiés incluent au moins l'un des paramètres suivants :

- ports local et distant
- numéro de référence local et distant
- numéro du prochain paquet à émettre et attendu
- horloge « timer » d'émission et de réception
- taille de fenêtre.

8. Procédé de migration selon la revendication 7, caractérisé en ce qu'il comprend en outre une sauvegarde de la liste des paquets en attente d'émission, des paquets qui sont en transit et des paquets reçus mais non encore lus par l'application.

9. Procédé pour réaliser une continuité de fonctionnement d'une application logicielle dans une architecture multi-ordinateurs (cluster), cette application étant exécutée à un instant donné sur l'un des ordinateurs du cluster, appelé nœud principal, les autres ordinateurs dudit cluster étant appelés nœuds secondaires, ce procédé mettant en œuvre le

procédé de migration selon l'une quelconque des revendications précédentes,
caractérisé en ce qu'il comprend les étapes suivantes :

- maintien au fil de l'eau d'au moins un clone de l'application sur au moins un des nœuds secondaires,
- en cas de détection d'une défaillance ou d'un événement affectant ledit nœud principal, basculement de service vers l'un au moins desdits clones, et
- migration de connexions réseau.

10. Procédé de continuité de fonctionnement selon la revendication 9, caractérisé en ce qu'il comprend en outre une mise à jour des clones de l'application.

11. Procédé de continuité de fonctionnement selon la revendication 10, caractérisé en ce que la mise à jour des clones de l'application est périodique.

12. Procédé de continuité de fonctionnement selon l'une des revendications 12 ou 11, caractérisé en ce que la mise à jour des clones de l'application est déclenchée sur un ou plusieurs événements caractéristiques.

13. Procédé de continuité de fonctionnement selon l'une des revendications 9 à 12, caractérisé en ce qu'il comprend en outre une supervision de l'état de ressources nécessairement au fonctionnement de l'application.

14. Procédé de continuité de fonctionnement selon l'une des revendications 9 à 13, caractérisé en ce qu'il comprend en outre, à la suite d'une détection d'une défaillance ou d'un événement affectant le nœud principal, une étape pour élire, parmi des clones installés sur des nœuds secondaires, un

clone pour être substitué à l'application initiale, le nœud sur lequel ledit clone élu est installé devenant le nouveau nœud principal.

15. Procédé de continuité de fonctionnement selon l'une des revendications 9 à 14, caractérisé en ce qu'il comprend en outre un enregistrement sur chaque clone de messages reçus par le nœud primaire, ces messages étant réinjectés dans le clone élu nouveau primaire en cas de basculement.

16. Système multi-ordinateurs prévu pour exécuter sur au moins desdits ordinateurs au moins une application logicielle, implémentant le procédé pour réaliser une continuité de fonctionnement selon l'une quelconque des revendications 9 à 15.

17. Application du procédé de migration de connexions selon l'une quelconque des revendications 1 à 8, pour une optimisation automatique de ressources informatiques par partage de charge par répartition dynamique de processus.

18. Application du procédé de migration de connexions selon l'une quelconque des revendications 1 à 8, pour une maintenance non interruptive par relocation à la demande de processus au travers d'un réseau de ressources informatiques.

19. Application du procédé de migration de connexions selon l'une quelconque des revendications 1 à 8, pour une préservation de contexte applicatif dans des applications mobiles.

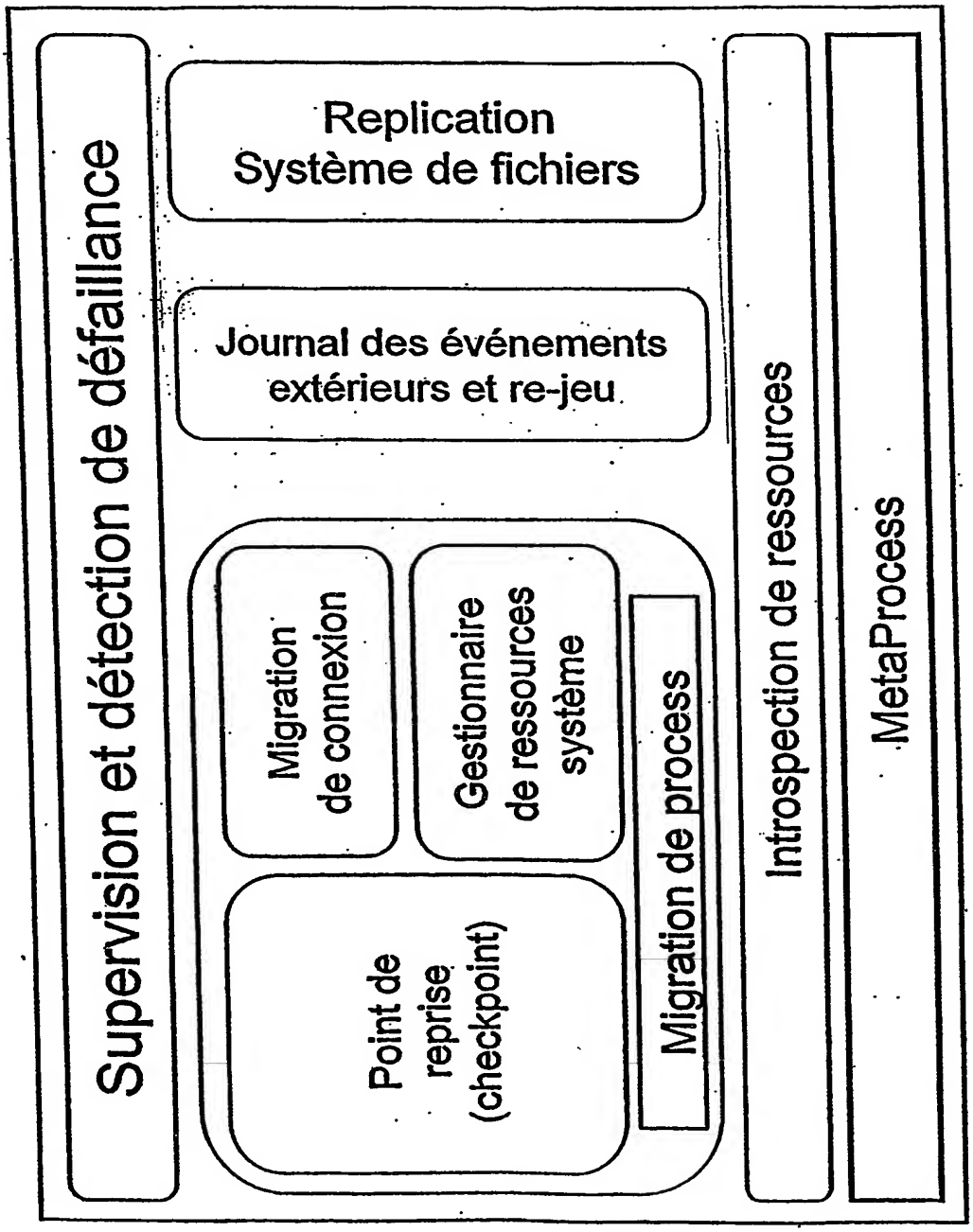
Supervision et détection de défaillance
Replication
Système de fichiers
Journal des événements
extérieurs et re-jeu
Migration
de connexion
Gestionnaire
de ressources
système
Migration de process
Point de
reprise
(checkpoint)
Introspection de ressources
MetaProcess

FIG.1

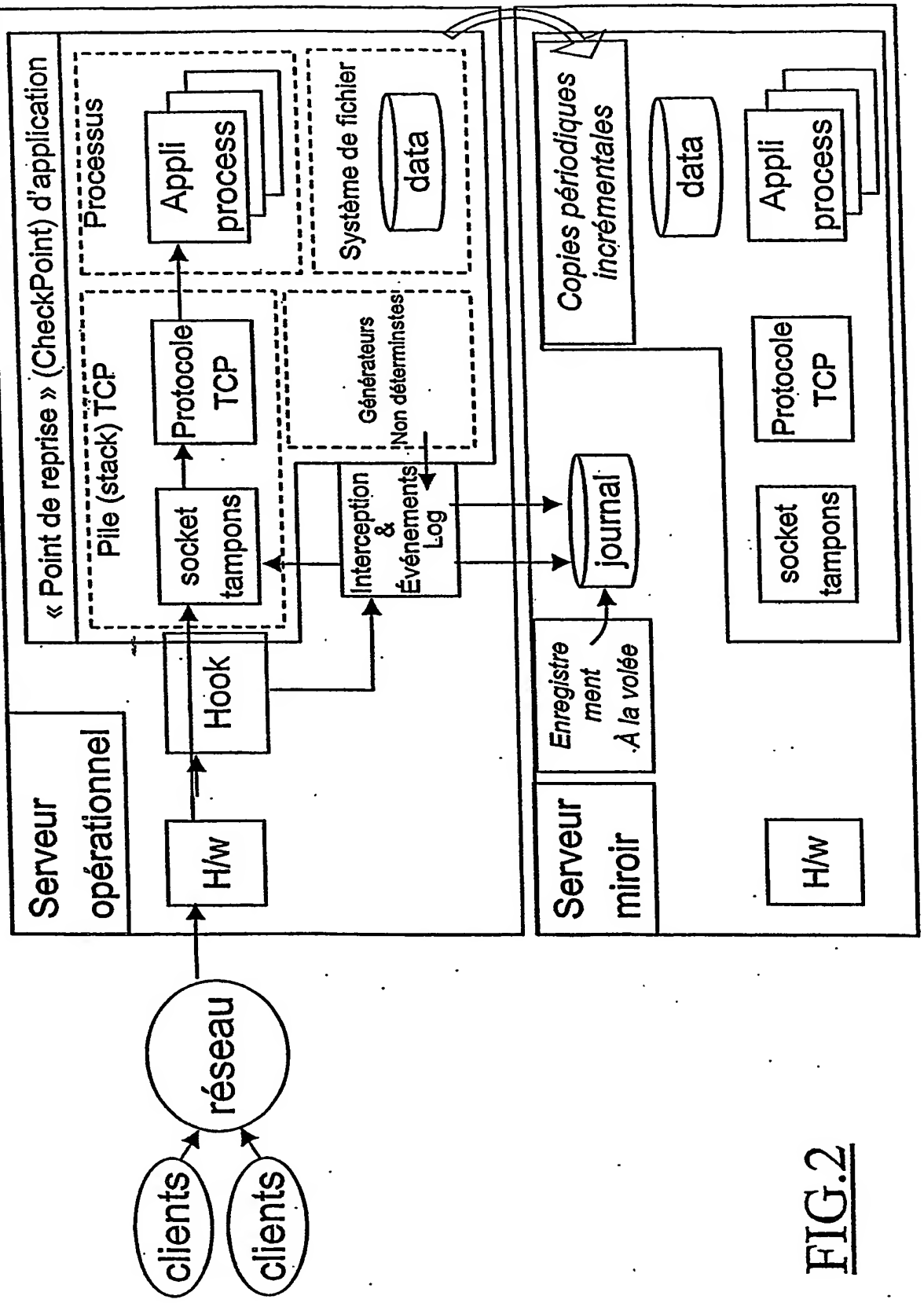
clients
clients
réseau
Serveur opérationnel
H/w
Hook
« Point de reprise » (CheckPoint) d'application
Pile (stack) TCP
socket tampons
Protocole TCP
Processus
Appli process
Système de fichier
data
Générateurs Non déterministes
Interception & Événements Log
journal
Enregistre ment À la volée
Serveur miroir
H/w
socket tampons
Protocole TCP
Appli process
data
Copies périodiques incrémentales

FIG.2

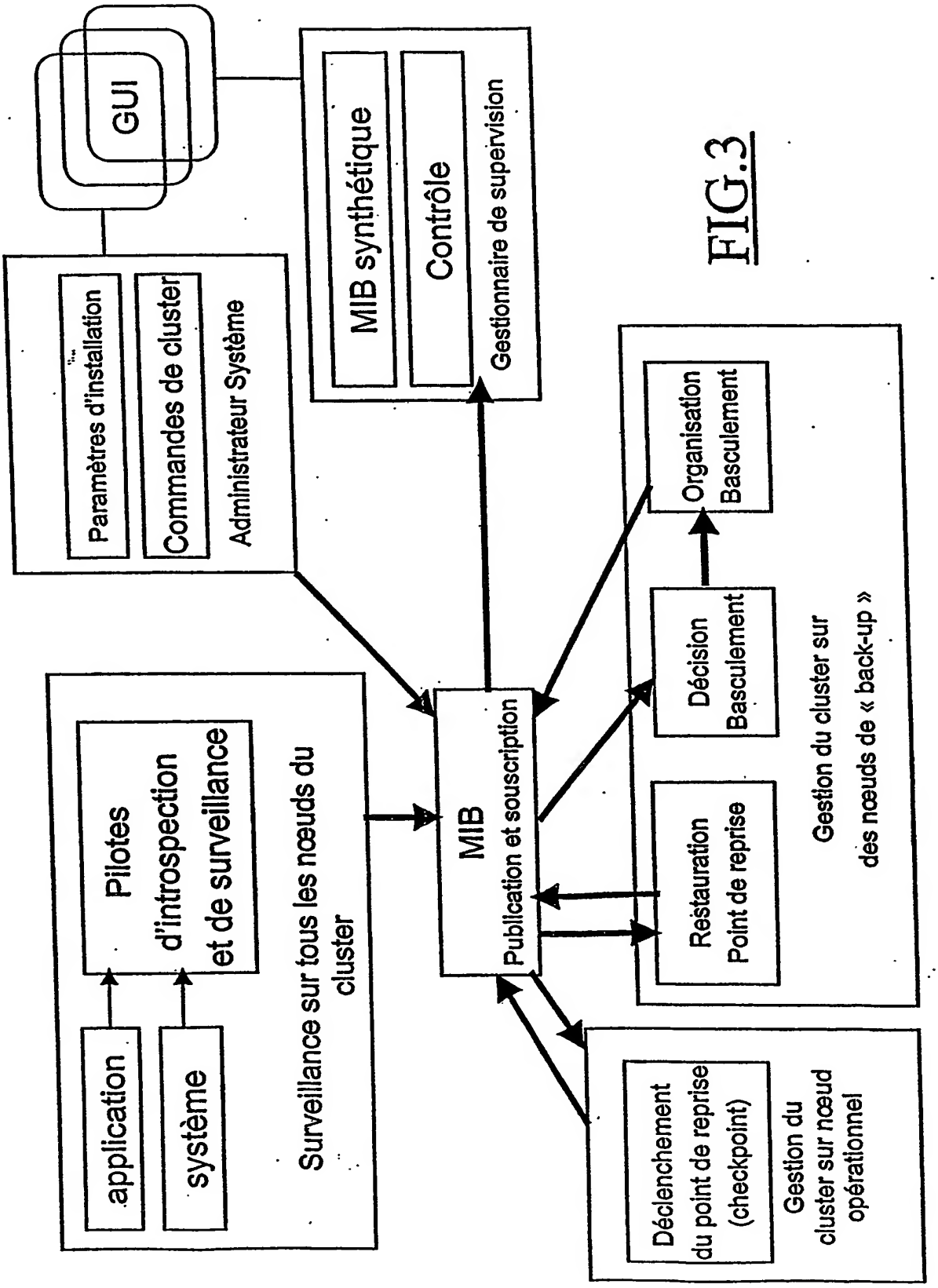
GUI
Paramètres d'installation
Commandes de cluster
Administrateur Système
MIB synthétique
Contrôle
Gestionnaire de supervision
Organisation Basculement
Décision Basculement
Restauration Point de reprise
Gestion du cluster sur des nœuds de « back-up »
Pilotes d'introspection et de surveillance
application
système
Surveillance sur tous les nœuds du cluster
MIB
Publication et souscription
Déclenchement du point de reprise (checkpoint)
Gestion du cluster sur nœud opérationnel

FIG.3